UPMC iCub project

From Wiki for RobotCub and Friends

Contents

Information about this guide

This guide is essentially a collection of several wiki pages to support the activities around the iCub in ISIR, UPMC. It provides instructions for the robot maintenance and its configuration, for developers and users. It is complementary to the official manual of iCub, and contains details of some configurations specific to iCubParis.

The guide is written and maintained by Serena Ivaldi (http://chronos.isir.upmc.fr/~ivaldi/) .

Please write to Serena for any problem you may encounter in following the instructions on these pages! The robot configuration and its cluster is in continuous evolution!

MACSi Project

The MACSi Project (http://macsi.isir.upmc.fr/) is a developmental robotics project based on the iCub humanoid robot. It is funded an as ANR Blanc project from 2010 to 2012.

- Macsi software repository (http://wiki.icub.org/wiki/UPMC_iCub_project/MACSi_Software)
- Macsi software documentation (http://chronos.isir.upmc.fr/~ivaldi/macsi/doc/) (work in progress)

Technical information for the iCub users in ISIR Lab

Cluster configuration

- configuration: here (http://wiki.icub.org/wiki/UPMC_iCub_project/MACSi_cluster)
- script files and demos: here (http://wiki.icub.org/wiki/UPMC_iCub_project/MACSi_scripts)
- libraries and environment variables: here (http://wiki.icub.org/wiki/UPMC_iCub_project/libraries)

The MACSi Project

Installing iCub on your pc

To install all the software you need for developing, running the simulator, and eventually test your applications on the robot, just follow the instructions on the iCub Manual (http://wiki.icub.org/wiki/Manual) .

- A more "verbose" guide to the installation, with pointers to the manual, is here (http://wiki.icub.org/wiki/UPMC_iCub_project/Short_guide_to_installation)
- A short guide for installing iCub simulator in Ubuntu: pdf (http://www.coboslab.psychologie.uni-wuerzburg.de/fileadmin/ext00209/user_upload/Publications/2011/2011Stalph-icub.pdf)
- Installing other libraries for iCub: here (http://wiki.icub.org/wiki/UPMC_iCub_project/Installing_libraries)

Using iCub

- starting iCub: here (http://wiki.icub.org/wiki/UPMC_iCub_project/Starting_iCub)
- using cameras: here (http://wiki.icub.org/wiki/UPMC_iCub_project/cameras)

iCub Simulators

- the official simulator of iCub based on ODE: installation (http://wiki.icub.org/wiki/ICub_Simulator_Installation)
- simulator based on python (Arboris-Python): installation (http://wiki.icub.org/wiki/UPMC_iCub_project/Arboris-Python) and first steps
- simulator based on XDE: installation (http://wiki.icub.org/wiki/UPMC_iCub_project/XDE-simulator) and first steps

iCub diary

If you want to know what happened to the iCub during his first days in ISIR, you can read this page (http://wiki.icub.org/wiki/UPMC_iCub_project/iCub_diary) . The page is no longer maintained because now too many things happen :)

Note: a software/hardware log is kept on the desk close to the power supplies. This

log/diary is used to keep track of updates (software, firmware, cluster config, ..), hardware/software issues and failures. If you notice something weird on iCub, or a failure happens, write down on it and then mail Serena (http://chronos.isir.upmc.fr/~ivaldi/) immediately!

Events

- InnoRobo 2013. iCub videos were shown in GDR-Robotique's stand in Innorobo, Lyon, on March 2013.
- AERES visit 2012. iCub was shown to the AERES committee with a live demo on November 2012.
- Fete de la Science 2012. iCub performed for two days in October, during the Science festival organised by ISIR-UPMC
- Fete de la Science 2011. iCub performed for two days in October, during the Science festival organised by ISIR-UPMC
- CLAWAR 2011. iCub will be the focus of a workshop organized jointly with Clawar (see the website (http://clawar2011.isir.upmc.fr/index.php?z=8&perma=workshop_icub))
- VES 2008. iCub in the ISIR-UPMC stand at the European city of Science (http://www.villeeuropeennedessciences.fr/uk/index.htm) expo, in Paris (see the video (http://vpadois.free.fr/envrac/iCub_movie_ves08_medium_res.avi) - with French comments)
- ICT 2008. iCub meets its cousin from IIT in Lyon

Pictures

Here (http://macsi.isir.upmc.fr/index.php?perma=1313440321) some pictures of iCub!

Videos

YouTube channel - iCubParis (http://www.youtube.com/user/iCubParis/videos?view=0)

Acknowledgments for this guide: Serena Ivaldi (http://chronos.isir.upmc.fr/~ivaldi/)

The old page (before may 2011) is still available here: here (http://wiki.icub.org/wiki/UPMC_iCub_project/run_icub)

UPMC iCub project/MACSi cluster

From Wiki for RobotCub and Friends

Contents

Latest news/updates

30/05/2013: **Cluster upgrade** - the cluster machines are upgraded from Ubuntu 10.04 to Ubuntu 12.04 LTD

General description

Network configuration

ICUB-GATEWAY is the server-gateway pc used to manage the cluster and the iCub@ISIR network. The following machines are available

```
10.0.0.254     : icub-gateway
10.0.0.1       : icubsrv
10.0.0.2       : pc104
10.0.0.11      : macsi01
10.0.0.12      : macsi02
10.0.0.13      : macsi03
10.0.0.14      : macsi04
```

The pc104 is on the robot, whereas icubsrv is a Ubuntu 12.04 (was 10.04 before the upgrade) server, sharing /icub/home/software and /exports. Dynamic IPs range from 150 to 160. Special static IPs can be set (ask the administrator = Serena).

Shared iCub home

You can access the precompiled iCub software by downloading the shared folder

```
icubsrv:/home/icub/software
```

using nfs. The folder has /lib, /share and /bin, containing yarp, icub and icub_isir applications and modules, and /src folder with the source code of all libraries. A bashrc_icub file is also provided, so you can add the environmental variables you need in your bashrc. All the computers of the cluster in the iCubParis01 network mount the same home directory from the server, so that the code is consistent everywhere. All computers being Ubuntu 12.04 (10.04 before), they mount the shared icub software folder in their /home/icub/software folder (note that Ubuntu users in their home also have pictures, videos etc).

Shared pc104 code

You can access the pc1404 code here

```
icubsrv:/exports/code-pc104
```

yarpserver

Generally, icubsrv hosts the yarp server (10.0.0.1 10000) and the namespace is /iCubParis01.

roscore

Generally, ros master (launched through roscore) is on macsi01.

Extra libraries

macsi01

- libboost-dev
- ROS

Important notice for ROS users In order to install ROS fuerte, the configuration of macsi01 is different with respect to the other machines of the cluster. Precisely, we had to uninstall the previous versions of libeigen3-dev and pcl* libraries in order to install the ones which ROS require (which are differently listed in apt lists). For more details, see what we did to install ROS fuerte (http://wiki.icub.org/wiki/UPMC_iCub_project /Installing_libraries#ROS) .

How to load iCub home into your computer

Hereinafter, we assume you have a freshly installed Ubuntu 12.04 LTD x64, with a user named 'icub', in the iCubParis01 network.

Note!

The following instructions were written for **Ubuntu 10.04 LTD x64**, they should be valuable for other versions as well: if not, and you encounter problems, please ask Serena Ivaldi (http://chronos.isir.upmc.fr/~ivaldi/contact.htm) for assistance.

You may start with getting ssh

```
sudo apt-get install ssh openssh-server
```

In order to mount its shared software folder, you must download (if you don't have it already) nfs:

```
sudo apt-get install nfs-common
```

you can now check if you can see the shared folders, for example you can use showmount and if everything is properly configured (i.e. you are within the iCubParis01 network) you should see something like this:

```
icub@macsi02:~$ showmount -e icubsrv
Export list for icubsrv:
/home/icub/software 10.0.0.0/24
/exports            10.0.0.0/24
```

All the computers in the cluster mount the shared folder in /home/icub/software. We recommend to mount it in the home folder of your icub user, so you don't have to modify the bashrc file we provide and recompile all the code. If you mount the shared folder in the icub's home directory, edit /etc/fstab

```
sudo nano /etc/fstab
```

and add this line at the bottom

```
icubsrv:/home/icub/software /home/icub/software nfs rw,intr,rsize=8192,wsize=8192 0 0
```

save and mount:

```
sudo mount -a
```

Important!

- **Note 1:** If you modify something in this folder, changes will reflect everywhere! So please, use this configuration only for new nodes in the cluster, and not for testing your code.
- **Note 2:** The code is continuously updated and compiled in icubsrv, so if you modify something and do not commit changes (using svn), you will loose everything at the following automatic update+recompile.

How to run iCub apps from your computer

Hereinafter, we assume you have a freshly installed Ubuntu 10.04 LTD x64, with a user named 'icub', and that you are correctly mounting /home/icub/software as explained in the previous section. Now you can modify your .bashrc file

```
cd
sudo nano .bashrc
```

by adding these few lines

```
# iCub software
if [ -f /home/icub/software/bashrc_icub ]; then
    . /home/icub/software/bashrc_icub
fi
```

or simply this line

```
source ~/software/bashrc_icub
```

These will automatically load the correct variables you need to run yarp and iCub modules and applications.

Important! To avoid known issues with yarprun (http://eris.liralab.it /wiki/Debugging_problems_with_yarprun) , if you intend to use the application GUIs to launch your programs (avoiding to launch each app with the command line) do not include these lines 'anywhere' in the .bashrc, but before the [-z "$PS1"] && return line, which is usually at the beginning of the .bashrc file. As an example, you should have:

```
# load icub environment vars even in non-interactive mode
# this is a bug-correction for yarprun
source ~/software/bashrc_icub
# If not running interactively, don't do anything
[ -z "$PS1" ] && return
```

To check if everything works, reload the terminal, and echo some variables, for example:

```
icub@macsi02:~$ echo $YARP_DIR
/home/icub/software/src/yarp2/build
icub@macsi02:~$ echo $ICUB_DIR
/home/icub/software/src/iCub/main/build
```

In /home/icub/software/src you can find yarp2, iCub, iCub_ISIR and some precompiled libraries:

- Ipopt (3.7.1 (http://www.coin-or.org/download/binary/Ipopt/Ipopt-3.7.1-linux-x86_64-gcc4.3.2.tgz))
- kdl (1.0.2 (http://people.mech.kuleuven.be/~rsmits/kdl/orocos-kdl-1.0.2-src.tar.bz2))
- lwpr (1.2.3 (http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/lwpr-1.2.3.zip))

- ode (0.11.1 (http://downloads.sourceforge.net/project/opende/ODE/0.11.1/ode-0.11.1.zip?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fopende%2Ffiles%2F&ts=1308741238&use_mirror=garr))
- OpenCV (2.2.0 (http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.2/OpenCV-2.2.0.tar.bz2?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fopencvlibrary%2Ffiles%2Fopencv-unix%2F2.2%2F&ts=1308740486&use_mirror=leaseweb))

The other libraries listed in the manual (http://eris.liralab.it/wiki/PrepareLinux) are missing, so you have to install them via apt-get

```
sudo apt-get update
sudo apt-get -y install cmake cmake-curses-gui g++ libncurses5-dev libace-dev libgtkmm-2.4-dev libglademm-2.4-
sudo apt-get -y install libqt3-mt-dev libgsl0-dev libsdl1.2-dev libglut3 libglut3-dev python-tk
```

Now you can run yarp and iCub modules.

If you need to run also iCub_ISIR code and MACSi code, you will need additional libraries, which are located in /home/icub/software/src:

- XCSF
- CUDA toolkit (4.0.17 (http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/cudatoolkit_4.0.17_linux_64_ubuntu10.10.run) - see NVIDIA website (http://developer.nvidia.com/cuda-toolkit-40))
- URBI

Important: to use CUDA, you have to install the drivers for your graphics card. In /home/icub/software/src you can find the linux drivers compatible with the CUDA version we provide. In some cases, if you have "nouveau" drivers installed (by default in Ubuntu), there could be problems in installing Nvidia dev-drivers directly; it is suggested to install Nvidia proprietary drivers first, and then the dev-drivers we provide.

Misc tips

- A quick link to pc104: create first a terminal profile named 'pc104', graphically different from the default (so running pc104 can be easily recognized) then add a link on the panel with

```
gnome-terminal --window-with-profile=pc104 -e "ssh -X pc104"
```

- Setting the yarp server:

```
yarp namespace /iCubParis01
yarp detect --write
```

- Remote password-less ssh: a very nice guide here (http://www.cyberciti.biz

/tips/linux-multiple-ssh-key-based-authentication.html)

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/MACSi_cluster&oldid=17265"

UPMC iCub project/MACSi scripts

From Wiki for RobotCub and Friends

Contents

Desktop scripts

iCub GUIs

Save the following text in a sh file (for example scriptLaunch.sh)

```
#!/bin/bash

cd $ICUB_ROOT/main/app/iCubCluster/scripts
./icub-cluster.py ../../robots/iCubParis01/scripts/cluster-config.xml &

cd $ICUB_ROOT/main/app/default/scripts
./manager.py ../../robots/iCubParis01/scripts/cameras.xml &
./manager.py ../../robots/iCubParis01/scripts/dumpData.xml &

cd $ICUB_ROOT/main/app
./icubapp.py app.txt &
```

then do

```
chmod a+x scriptLaunch.sh
```

now you can execute it to launch the main iCub GUIs.

Converting images from ppm to jpg (or other formats)

When dumping images from cameras with DataDumper, images are saved in .ppm format. To convert such images in jpg (or other formats, using ImageMagick *convert*) you can use this command

```
ls *.ppm | sed -e "s/.ppm$//" | xargs -n1 --replace convert -verbose {}.ppm {}.jpg
```

To create an animated gif for preview purposes (e.g. for a web page) you can select some images and using the same package

```
convert -delay 100 -loop 0 image*.jpg animation.gif
```

To create a video from single images do

```
ffmpeg -r 24 -b 2000000 -i %08d.ppm test.avi
```

Useful commands for images

Appending multiple images in the same row (es: Im1 Im2 Im3... ImN)

```
convert Im* +append ImROW.png
```

Server scripts

Update yarp and iCub on icubsrv and pc104

This script is located in the server (icubsrv), and is used to update yarp and iCub repositories for both pc104 and cluster. To launch it do:

```
ssh icubsrv
sh svn_update_everything.sh
```

Here is the content:

```
#!/bin/bash
echo "Starting svn update on icubsrv..."
echo "pc104 - yarp"
cd /exports/code-pc104/yarp2
svn update
echo "icubsrv - yarp"
cd /home/icub/software/src/yarp2
svn update
echo "pc104 - icub"
cd /exports/code-pc104/iCub
svn update
echo "icubsrv - icub"
cd /home/icub/software/src/iCub
svn update
echo "icubsrv - icub_isir"
cd /home/icub/software/src/iCub_ISIR
svn update
echo "icubsrv - icub_macsi"
cd /home/icub/software/src/iCub_MACSI
svn update
echo "icubsrv - icub_chris"
```

```
cd /home/icub/software/src/iCub_CHRIS
svn update
echo "icubsrv - icub_italk"
cd /home/icub/software/src/iCub_ITALK
svn update
echo "Finished svn update on icubsrv!!"
```

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/MACSi_scripts&oldid=17008"

UPMC iCub project/libraries

From Wiki for RobotCub and Friends

Contents

Linux

iCub modules

If you are loading iCub software from the cluster server (http://eris.liralab.it /wiki/UPMC_iCub_project/MACSi_cluster) , in /home/icub/software/src you can find yarp2, iCub and some precompiled libraries:

- Ipopt (3.7.1 (http://www.coin-or.org/download/binary/Ipopt/Ipopt-3.7.1-linux-x86_64-gcc4.3.2.tgz))
- ode (0.11.1 (http://downloads.sourceforge.net/project/opende/ODE/0.11.1 /ode-0.11.1.zip?r=http%3A%2F %2Fsourceforge.net%2Fprojects%2Fopende%2Ffiles%2F&ts=1308741238& use_mirror=garr))
- OpenCV 2.2.0 (2.2.0 (http://downloads.sourceforge.net/project/opencvlibrary /opencv-unix/2.2/OpenCV-2.2.0.tar.bz2?r=http%3A%2F %2Fsourceforge.net%2Fprojects%2Fopencvlibrary%2Ffiles%2Fopencv-unix%2F2.2%2F&ts=1308740486&use_mirror=leaseweb))

The other libraries listed in the manual (http://eris.liralab.it/wiki/PrepareLinux) are missing, so you have to install them via apt-get

```
sudo apt-get update
sudo apt-get -y install cmake cmake-curses-gui g++ libncurses5-dev libace-dev libgtkmm-2.4-dev libglademm-2.4-
sudo apt-get -y install libqt3-mt-dev libgsl0-dev libsdl1.2-dev libglut3 libglut3-dev python-tk libqwt5-qt3-dev
```

Important update (sept-12): other libraries have been added, the list is becoming quite big. also, some libraries have changed name from previous versions of ubuntu. A

quite convenient solution is to install icub-common, which includes all dependencies. On Ubuntu 12.04 for example do:

- configure your apt to look for the correct list as described here (http://eris.liralab.it /wiki/Linux:Installation_from_binaries) , for example in Ubuntu 12 do:

```
sudo sh -c 'echo "deb http://www.icub.org/ubuntu precise contrib/science" > /etc/apt/sources.list.d/icub.list'
```

- then install icub-common

```
sudo apt-get install icub-common
```

- you may want to check here (http://eris.liralab.it /wiki/Linux:Installation_from_binaries)

iCub_ISIR modules

If you need to run iCub_ISIR code, you will need additional libraries, which are located in /home/icub/software/src:

- kdl (1.0.2 (http://people.mech.kuleuven.be/~rsmits/kdl/orocos-kdl-1.0.2-src.tar.bz2))
- lwpr (1.2.3 (http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/lwpr-1.2.3.zip))
- eigen2 (required for kdl (http://eigen.tuxfamily.org/index.php?title=Main_Page))
- orocos rtt (required for kdl (http://www.orocos.org/rtt/subversion))
- boost (required by orocos rtt (http://www.boost.org/users/download/))
- stereo
- cold
- qhull 2011.1
- OpenCV 2.3.1

Other libraries can be installed via apt-get

```
sudo apt-get -y install libglew1.5-dev libblas-dev liblapack-dev
sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl
sudo apt-get update
sudo apt-get install libpcl-all
```

More info about:

- GLEW here (http://glew.sourceforge.net/index.html)
- PointCloud here (http://pointclouds.org/downloads/linux.html)

MACSi modules

- URBI (urbi 2.7.5 (http://www.gostai.com/downloads/urbi/2.7.5/urbi-sdk-2.7.5-

linux_lucid-x86-gcc4.tar.bz2) and urbi-yarp 1.0 (http://www.gostai.com/downloads/yarp/urbi-yarp-sources-1.0.zip))
- CUDA toolkit (4.0.17 (http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/cudatoolkit_4.0.17_linux_64_ubuntu10.10.run) - see NVIDIA website (http://developer.nvidia.com/cuda-toolkit-40))
- pae (svn with authentication (https://scm.gforge.inria.fr/svn/pae/trunk)) required for vision modules

- Kinect

```
sudo apt-get install openni-dev ps-engine
sudo apt-get install libboost-all-dev libusb-1.0-0-dev libqt4-dev libgtk2.0-dev cmake libglew1.5-dev libgsl0-d
sudo apt-get install libcminpack-dev
```

to install OpenNI, download ROS version (as suggested in the CMakeLists for YARP's kinect driver)

```
sudo apt-get install mercurial git-core doxygen
hg clone https://kforge.ros.org/openni/drivers OpenNI
cd OpenNI
make
```

Note: OpenNI is not necessary if ROS is installed.

If you're using the kinect from OpenCv, then OpenNI must be used in combination with the PrimeSense driver, as explained in OpenCV guide here. (http://opencv.itseez.com/doc/user_guide/ug_highgui.html)

Environment variables

If you are loading iCub software from the cluster server (http://eris.liralab.it/wiki/UPMC_iCub_project/MACSi_cluster) , there's a bash file called **bashrc_icub**, with the environment variables you need:

```
echo "Exporting iCub environment variables from specific file.."
export ICUB_INSTALL_PREFIX=/home/icub/software
export OPT_INSTALL_PREFIX=/home/icub/software/opt
export YARP_ROOT=$ICUB_INSTALL_PREFIX/src/yarp2
export YARP_DIR=$YARP_ROOT/build
export YARP_CONF=/home/icub/.yarp
export ICUB_ROOT=$ICUB_INSTALL_PREFIX/src/iCub
export ICUB_ROBOTNAME=iCubParis01
export ICUB_DIR=$ICUB_ROOT/main/build
export ODE_DIR=$ICUB_INSTALL_PREFIX/src/ode
export OPENCV_DIR=$ICUB_INSTALL_PREFIX/src/OpenCV/build
export IPOPT_DWN_DIR=$ICUB_INSTALL_PREFIX/src/Ipopt
export IPOPT_DIR=$ICUB_INSTALL_PREFIX/src/Ipopt/build
export XCSF_DIR=$ICUB_INSTALL_PREFIX/src/XCSF
export ISIR_ROOT=$ICUB_INSTALL_PREFIX/src/iCub_ISIR
export ISIR_DIR=$ISIR_ROOT/build
export LIBSTEREO_LIBRARY_DIR=$OPT_INSTALL_PREFIX/lib
export LIBSTEREO_INCLUDE_DIR=$OPT_INSTALL_PREFIX/include
export QHULL_DIR=$ICUB_INSTALL_PREFIX/src/qhull/build
export QHULL_INCLUDE_DIRS=$OPT_INSTALL_PREFIX/include/libqhull
export QHULL_LIBRARY=$OPT_INSTALL_PREFIX/lib
```

```
export QHULL_LIBRARY_DEBUG=$OPT_INSTALL_PREFIX/lib/libqhull6.so
export OPENCV_NEW_DIR=$ICUB_INSTALL_PREFIX/src/OpenCV_new/build
export CUDA_DIR=$ICUB_INSTALL_PREFIX/cuda
export MACSI_ROOT=$ICUB_INSTALL_PREFIX/src/iCub_MACSI
export MACSI_DIR=$MACSI_ROOT/main/build
export PAE_ROOT=$ICUB_INSTALL_PREFIX/src/pae
export PAE_DIR=$PAE_ROOT
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/home/icub/software/lib/pkgconfig/
export CLASSPATH="$XCSF_DIR/JavaXCSF/xcsf.jar:$XCSF_DIR/xcsf-server/xcsfserver.jar:$XCSF_DIR/xcsfServer.jar"
export PATH=$PATH:$ICUB_INSTALL_PREFIX/bin:$ICUB_INSTALL_PREFIX/scripts:$CUDA_DIR/bin:$OPT_INSTALL_PREFIX/bin
export LD_LIBRARY_PATH=$ICUB_INSTALL_PREFIX/lib:$CUDA_DIR/lib64:$CUDA_DIR/lib:$IPOPT_DIR/lib:$OPT_INSTALL_PREFIX
echo "Welcome $ICUB_ROBOTNAME!"
```

You can easily include them in your .bashrc file (/home/icub/.bashrc) by appending these lines at the bottom:

```
# iCub software
if [ -f /home/icub/software/bashrc_icub ]; then
    . /home/icub/software/bashrc_icub
fi
```

As a test, open a terminal, and type "bash". You should get something like that:

```
icub@macsi03:~$ bash
Exporting iCub environment variables from specific file..
Welcome iCubParis01!
```

If you also have ROS on the machine, add these lines at the end of .bashrc file

```
echo "Exporting ROS variables.."
export ROS_WORKSPACE=/home/icub/software/src/ros_workspace
export ROS_PACKAGE_PATH=$ROS_WORKSPACE:$ROS_PACKAGE_PATH
export ROS_HOSTNAME=macsi01
export ROS_MASTER_URI=http://macsi01:11311/
echo "ROS master at $ROS_MASTER_URI"
```

Note that ROS_MASTER_URI is necessary to connect ROS and YARP, and in fact is the URI of roscore when it starts. It is here assumed that ROS is installed on macsi01 and it is only "locally" installed: that's why we don't put these environment variables in the shared bashrc_icub file. By the way, we noticed that if we put these variables at the beginning of the bashrc file, some ROS functions do not work (beware, ros seems quite variable in its behavior).

Windows

Hereinafter, I assume you are using Windows 7 and Visual Studio 10 (msvc10).

iCub modules

Follow the official manual (http://eris.liralab.it/wiki/PrepareWindows) . Precompiled modules are suggested.

iCub_ISIR modules

- kdl
- lwpr
- eigen2 (required for kdl)
- orocos rtt (required for kdl)

MACSi modules

- URBI
- CUDA toolkit

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/libraries&oldid=16348"

UPMC iCub project/Installing libraries

From Wiki for RobotCub and Friends

Contents

Linux

Note: these notes assume you are putting sources in /home/icub/software/src and installing everything in /home/icub/software (as in the icub cluster).

Boost

Boost is required by Orocos-Rtt. The official link is here (http://www.boost.org/users/download/) . Download Boost

```
svn co http://svn.boost.org/svn/boost/trunk boost
```

Launch the scripts to install

```
./bootstrap.sh --prefix=/home/icub/software
./b2
```

In .bashrc, add the environment variable **BOOST_ROOT** (required by orocos-rtt)

```
export BOOST_ROOT=$ICUB_INSTALL_PREFIX/src/boost
```

Alternatively, you can get the libraries via apt-get.

Orocos-Rtt

Orocos-Rtt is required by KDL. The official link is here (http://www.orocos.org/rtt/subversion) . Download Orocos-Rtt

```
svn co http://svn.mech.kuleuven.be/repos/orocos/trunk/rtt rtt
```

Follow the instructions here (http://www.orocos.org/stable/documentation/rtt/v1.12.x/doc-xml/orocos-installation.html#first-source-tree) to install, precisely

```
mkdir build
cd build
cmake
cmake .. -DOROCOS_TARGET=gnulinux -DCMAKE_INSTALL_PREFIX=/home/icub/software
make
make install
```

Eigen2

Eigen2 is required by KDL. The official link is here (http://eigen.tuxfamily.org/index.php?title=Main_Page) . Attention: download eigen2 and not eigen3. Download eigen2

```
wget http://bitbucket.org/eigen/eigen/get/2.0.15.tar.bz2
```

Extract it in /eigen2, enter and do:

```
mkdir build
cd build
ccmake ..
make
make install
```

KDL

Download kdl

```
wget http://people.mech.kuleuven.be/~rsmits/kdl/orocos-kdl-1.0.2-src.tar.bz2
```

Extract it in kdl, enter and follow the instructions in INSTALL, doing:

```
mkdir build
cd build
ccmake ..
make
make install
```

In .bashrc, add the environment variable **KDL_DIR** (required by iCub_MACSI code) and make it point to the path where you installed kdl (it will look for lib, include..), for

example

```
export KDL_DIR=$ICUB_INSTALL_PREFIX
```

LWPR

Download LWPR

```
wget http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/lwpr-1.2.3.zip
```

Extract it in lwpr, enter and follow the instructions in INSTALL.TXT, doing:

```
./configure
make
make install
```

PAE

Download pae via svn

```
svn checkout https://scm.gforge.inria.fr/svn/pae/trunk pae
```

Then in the folder do

```
./bootstrap
./configure --enable-opencv=yes --enable-surf=no --prefix=/home/icub/software
make
make install
```

If the configure script doesn't find opencv (e.g. because they are installed in a different directory than /usr/local/) add this line to .bashrc

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/home/icub/software/lib/pkgconfig/
```

where the directory on the left points toward the folder where the file **opencv.pc** has been installed by OpenCV during its installation. Note: you may need to install these packages

```
sudo apt-get install libtool automake
```

qhull

Download qhull

```
wget http://www.qhull.org/download/qhull-2011.1-src.tgz
```

Extract it, then

```
mkdir build
cd build
ccmake ..
make
make install
```

ROS

To install ROS fuerte, simply follow the installation guide on the official website (http://www.ros.org/wiki/fuerte/Installation/Ubuntu) . ROS will be installed in /opt/ros /fuerte. It is necessary then to modify the .bashrc file to load ROS' environment variables. For example in our bashrc we have:

```
# load icub environment vars even in non-interactive mode
source ~/software/bashrc_icub
# If not running interactively, don't do anything
[ -z "$PS1" ] && return
.....
# better to put this at the end of the file
# load ros environment vars
source /opt/ros/fuerte/setup.bash
.....
# ROS environment variables
....
```

Some environment variables have to be added manually to make ROS cohexist with YARP: see how to add these variables in the bashrc_icub (http://eris.liralab.it /wiki/UPMC_iCub_project/libraries#Environment_variables) . If you have an earlier installation of eigen3 or pcl, you may need to remove those libraries. In our case on Ubuntu 10.04 we did the following:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu lucid main" > /etc/apt/sources.list.d/ros-latest.list
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
sudo rm /etc/apt/sources.list.d/v-launchpad-jochen-sprickerhof-de-pcl-lucid.list*
sudo apt-get remove libeigen3-dev
sudo apt-get update
sudo apt-get install ros-fuerte-desktop-full
sudo apt-get install python-pip
sudo easy_install -U rosinstall vcstools rosdep
```

Generally, ROS requires a number of libraries, which you may want to install before, for example:

```
sudo apt-get install build-essential python-yaml cmake subversion wget python-setuptools mercurial git-core
sudo apt-get install python-yaml libapr1-dev libaprutil1-dev libbz2-dev python-dev libgtest-dev python-paramik
sudo apt-get install python-wxgtk2.8 python-gtk2 python-matplotlib libwxgtk2.8-dev python-imaging libqt4-dev g
```

You can verify that ROS is installed correctly by typing for example:

```
rosversion ros
rospack find <name of basic package>
```

To enable YARP-ROS communication, follow the instructions on YARP's website (http://eris.liralab.it/yarpdoc/yarp_with_ros.html) .

URBI

To install Urbi, download the sdk or the precompiled version. If you choose to install the sdk and compile yourself do:

```
wget http://www.gostai.com/downloads/urbi/2.7.5/urbi-sdk-2.7.5.tar.bz2
ln -s urbi-sdk-2.7.5 urbi-root
```

Then follow the instructions for building on the website (http://www.gostai.com /downloads/urbi/doc/build.html) . Urbi has many dependencies, so before starting you may need to get some libraries, e.g.

```
sudo apt-get install aspell aspell-en autoconf automake bc ccache colordiff coreutils cvs doxygen flex g++ get
```

Many of these libraries should be already installed in your machine. It is also possible to get a precompiled version of urbi. For our Lucid we did:

```
wget http://www.gostai.com/downloads/urbi/2.7.5/urbi-sdk-2.7.5-linux_lucid-x86-gcc4.tar.bz2
```

Unpack it in a user-accessible folder and rename it as urbi-root. Follow the instructions to install. Then add the environment variable URBI_ROOT pointing at this folder, and add its /bin folder to the PATH. For example do something like:

```
export URBI_ROOT=$ICUB_INSTALL_PREFIX/urbi-root
export PATH=$PATH:$ICUB_INSTALL_PREFIX/bin:$URBI_ROOT/bin: <the rest>
```

You now need to install the yarp-urbi bridge.

```
wget http://www.gostai.com/downloads/yarp/urbi-yarp-sources-1.0.zip
unzip urbi-yarp-sources-1.0.zip
cd urbi-yarp
wget http://www.gostai.com/downloads/tools/FindUrbi.cmake
```

The standard instruction to compile this package is:

```
umake-shared -o yarp yarp.cc -I/<PathToYarp>/include -L/<PathToYarp>/lib -lYARP_dev -lYARP_OS -lYARP_sig -lYAR
```

which in our case (where everything is installed in /home/icub/software) is:

```
umake-shared -o yarp yarp.cc -I/home/icub/software/include -L/home/icub/software/lib -lYARP_dev -lYARP_OS -lYA
```

HARK

Hark is a library for sound source detection and localization. We use it with a set of ROS modules (hence it is only installed in macsi01). Ti install it, follow the guidelines here (http://winnie.kuis.kyoto-u.ac.jp/HARK/wiki.cgi?page=HARK+Installation+Instructions) . Do:

```
sudo sh -c 'echo "deb http://winnie.kuis.kyoto-u.ac.jp/HARK/harkrepos lucid non-free\ndeb-src http://winnie.ku
wget -q -O - http://winnie.kuis.kyoto-u.ac.jp/HARK/harkrepos/public.gpg | sudo apt-key add -
sudo apt-get update
sudo apt-get install harkfd
sudo apt-get install harktool3
```

For ROS fuerte users also do this:

```
sudo apt-get install harkfd hark-ros-fuerte hark-ros-stacks-fuerte
```

this is for ros-electric

```
sudo apt-get install harkfd hark-ros-electric hark-ros-stacks-electric
```

To install the supported driverss follow the link on the website. For example for kinect (http://winnie.kuis.kyoto-u.ac.jp/HARK/wiki.cgi?page=HARK-KINECT+Installation+Instructions) .

Python Binding for Yarp (on OS X)

Install SWIG :

```
brew install swig
```

Giving that $YARP_DIR is where you svn checkout is:

```
cd $YARP_DIR/bindings
mkdir build
cd build
ccmake ..
```

Check CREATE_PYTHON, press 'c' twice, 'g' once.

```
make
```

If you run make install, it will probably put them in the wrong place. So, assuming you are using python brew :

```
cp yarp.py ~/.pythonbrew/pythons/Python-2.7.3/lib/python2.7/site-packages/
cp _yarp.so ~/.pythonbrew/pythons/Python-2.7.3/lib/python2.7/site-packages/
```

To verify all is well, run

```
python
```

And type 'import yarp', and press enter. If nothing happens, all is good.

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/Installing_libraries&oldid=16317"

UPMC iCub project/Short guide to installation

From Wiki for RobotCub and Friends

This short guide will help you in the installation of YARP and ICUB in your own machine: its content is basically a more "verbose" description of the steps you can find in the iCub Manual (http://eris.liralab.it/wiki/Manual) . Here you will find the shortcuts to the main steps, with some simple notes. To run the iCub simulator and start developing in YARP for the robot, you need to install first YARP then the iCub software, following the installation guide here (http://eris.liralab.it /wiki/Manual#Six._Software.2C_Compiling_YARP_and_iCub) . For doing this, you need to "prepare" your machine, and depending on the operative system you're using (linux or windows) it will take a while.

The basic steps to get "yarped" and "icubbed"

- **Install libraries**. There is a list of libraries to install (GTK,GSL,ACE,IPOPT,OPENCV,SDL,QT and so on); which are required to compile Yarp and the iCub software. There are precompiled packages which are recommended if you're on Windows; whereas on Linux just follow the instructions to get the lib packages.
  - libraries in windows (http://eris.liralab.it/wiki/PrepareWindows)
  - libraries in linux (http://eris.liralab.it/wiki/PrepareLinux)

It is usually recommended to put all the code you need in a single folder, like *C:* to have: *C:\yarp*, *C:\icub*, *C:\ace* ... and so on. This will facilitate browsing and updating code if it's the first time you use the iCub.

- **Environment variables**. Remember to create the environmental variables for every library you install, because they are necessary to compile everything. Once you've installed all the libraries (GTK,GSL,ACE,IPOPT,OPENCV,SDL,QT and so on), you can finally get and compile yarp and iCub software.

- **Svn client**. First, you need a subversion client to download the packages: see here (http://eris.liralab.it/wiki/Getting_Subversion) .

- **Get Yarp and iCub with svn**. Use the instructions on the manual to get the yarp and iCub code. For example, for yarp you must go into the yarp folder (e.g. *C:\yarp*), link the folder to the svn repository, and update to get the code:
  - get Yarp (http://eris.liralab.it/wiki/GettingYARP_svn)
  - get iCub (http://eris.liralab.it/wiki/Getting_the_iCub_software)

- **CMake**. Once you're done with the download, you need CMake to create the software projects (to support cross-platform compilation) and compile both projects in your machine: see here (http://eris.liralab.it/wiki/CMake_icub) .

- **Compile yarp**. Go into the YARP folder first, and compile Yarp: first, run CMake to create the project, then compile. There's a detailed tutorial on YARP, with more information here (http://eris.liralab.it/wiki/YARP_Tutorial) .

- **Check yarp**. After the compile and installation procedure, you may want to check if it is installed correctly.

Basically, once you've compiled yarp in your system, just open a terminal shell or a command window, and type "*yarp*" if it is installed correctly it should start saying "*This is the YARP network companion*". Then type "*yarp help*" to see the available list of commands.

- **Compile icub**. If you're "yarped", you can go on compiling iCub. Go into the iCub folder (e.g. *C:\icub*), run CMake, compile.

Once you're done, you can start playing with the iCub simulator to see if everything is fine. More details about the simulator can be found here (http://eris.liralab.it /wiki/Simulator_README) .

- **Launch simulator**. Steps are:
  - open a terminal and launch the yarp server, type "*yarp server*"
  - open a terminal and launch the iCub simulator, type "*iCub_SIM*"
  - open a terminal and type "*yarp name list*" to see all the ports opened by yarp. If the simulator is running, some ports related to the iCub and its state must exist
  - you can now start playing with the simulator.

- **Applications**. One more thing to do is to enable the automated running of applications with the GUIs: see here (http://eris.liralab.it /wiki/Running_applications) . The GUIs are python-based, so you need to have python and prepare your system for running applications (http://eris.liralab.it /wiki/Prepare_your_system_for_running_applications) .

I want to know more...

For more details please refer to the complete iCub Manual (http://eris.liralab.it /wiki/Manual) .

I want to use the robot...

If you want to use the robot, read the manual (http://eris.liralab.it/wiki/Manual) first, and particularly:

- starting the icub (http://eris.liralab.it/wiki/ICub_startup)
- checking the iCubInterface (http://eris.liralab.it/wiki/ICubInterface_Errors)

Then refer to this page (http://eris.liralab.it/wiki/UPMC_iCub_project/run_icub) for specific details for the configuration of the iCub network in ISIR Lab.

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/Short_guide_to_installation&oldid=12552"

UPMC iCub project/MACSi Software

From Wiki for RobotCub and Friends

Contents

Getting MACSi software

You can get MACSi code using a subversion client (Linux = svn ; Windows = tortoiseSvn) and the following command:

```
svn co https://hotline.isir.upmc.fr/svn/macsi/
```

Currently, the repository is secured by login and password. To obtain yours, ask Serena (serena.ivaldi _AT_ isir.upmc.fr).

Setting up your system

To compile MACSi code, you need to set up your machine properly: particularly, you must install YARP and iCub and prepare your system with libraries and environment variables (http://eris.liralab.it/wiki/UPMC_iCub_project/libraries) .

Notes:

- some modules require ROS.
- some vision modules require PAE and OpenNI. PAE is a vision library developed within INRIA, and the access is not open yet. If you're running into troubles compiling vision modules, simply uncomment the modules in the CMakelists.
- some vision modules require different versions of OpenCV. We suggest you to install the other OpenCV versions in custom locations (for example /home/icub/opt) to avoid conflicts, then check the makefile to verify that each module is linking the proper version.
- Some CMake files of iCub have been recently changed particularly **iCubFindDependencies.cmake** is no longer installed. This file is used by MACSi's CMake files to keep MACSi code aligned with iCub. In MACSi's CMake main file it is evoked by default, so you will need to have it installed manually (or you can edit

iCub's CMake file). This procedure is not necessary, but warmly encouraged to facilitate the compiling/linking.

If you have questions or troubles, please write to Serena Ivaldi (http://chronos.isir.upmc.fr/~ivaldi/) .

Compiling

The repository has the same organization of iCub, indeed exploits some CMake files to create the project and its executables. The procedure is the following:

- Install YARP
- Install iCub
- Check the installation of the CMake files, praticularly iCubFindDependencies.cmake
- Get MACSi code via svn
- Set the following environment variables (assuming you put the code in $CODE/macsi) :

```
$MACSI_ROOT = $CODE/macsi
$MACSI_DIR = $CODE/macsi/main/build
```

- Create MACSI_DIR, for example

```
cd $MACSI_ROOT/main
mkdir build
```

- Compile MACSi, creating a CMake project

```
cd $MACSI_DIR
ccmake ..
make
make install
make install_applications
```

Now you can run MACSi modules!

Notes:

- In the iCub cluster at ISIR, we put the code in */home/icub/software /src/iCub_MACSI*
- You may want to change the INSTALLATION_PREFIX during the CMake configuration, if you want to put the executables in a specific folder. Remember to add this foder to the system path. For example in the iCub cluster we put everything in /home/icub/software -> /bin /lib /include /share

Running applications

You can run modules both from command-line and from the application GUI.

Configuration files are generally stored in $MACSI_ROOT/main/app. Each module or application has its folder (e.g. module) and there's a standard for its configuration files:

```
$MACSI_ROOT/main/app/module/scripts
```

stores a *.xml.template file, which is used to provide a template for the xml file used to launch the application GUI. Copy the xml template file and rename it as .xml (i.e. remove the .template) and edit it to fit it with your configuration, particularly to comply with the yarprun servers.

```
$MACSI_ROOT/main/app/module/conf
```

stores the configuration files, which are generally used to read specific parameters (e.g. the robot = icub or icubSim) for the module. You can create yours to change parameters.

Command line

- Launch module:

```
module --from $MACSI_ROOT/main/app/module/conf/file.ini
```

- Connect ports manually

```
yarp connect /module/port:o /others_module/port:i
...
yarp connect /others_module/port:o /module/port:i
```

- To terminate the module, first disconnect ports manually
- Send terminate commands where possible:

```
yarp terminate /module
```

- Use ctrl+c

Using the GUI

- Launch GUI using python manager

```
manager.py $MACSI_ROOT/main/app/module/scripts/module.xml
```

- Click on "run modules" then "connect" to run the module
- To terminate the module, click on "disconnect" then "stop modules"

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/MACSi_Software&oldid=17149"

ICub Simulator Installation

From Wiki for RobotCub and Friends

iCub Simulation

Contents

Requirements

Parts of the RobotCub software and the iCub simulator depend on external libraries. Here we keep track of the list of libraries needed and documentation for installing them.

The typical RobotCub software dependencies, see Chapter 6 of the iCub manual (http://eris.liralab.it/wiki/Manual) for instructions on how to install these dependencies.

The iCub Simulator needs extra libraries:

- ODE (Open Dynamic Engine)
- SDL (Simple DirectMedia Layer)

Installation

We assume you have prepared your system as in the Chapter 6 of the manual (http://eris.liralab.it/wiki/Manual).

Important: the iCub installation binaries provide all the dependencies and the simulator so you no longer need to follow the steps below. To install the simulator you can follow the iCub installation instructions in the manual.

ODE

Instructions for different systems are reported below.

Common notes:

The simulator now supports version >= 0.10built as single or double precision. Double precision is recommended and important if planning to import 3D models into the simulator.

Single precision is much faster and uses less memory but you might encounter more numerical errors. The double precision was suggested as you would have more accuracy and stability, the downside is that it works a bit slower.

Linux

sudo apt-get install libglut3 libglut3-dev

Linux: Installing ODE

Windows: binaries

- We provide binaries for Microsoft Visual Studio 2008 and 2009 at:

```
http://eris.liralab.it/iCub/downloads/packages/windows/common
```

Just unpack the zip file.

Set the environment variable ODE_DIR to point to the directory where you unpacked the package.

Windows: compilation

Follow these instructions if you do not want to use precompiled binaries (as described above), but prefer to compile ode on your system.

- Get the sources from the ODE website.

```
Generate project files for your compiler:
 cd %ODE_DIR%/build
 premake --target vs2005
```

To get a list of targets run premake --help

- Open the project file in %ODE_DIR%/build and compile. In windows the library **should be compiled in all modes, release, debug single and double prevision**. If you wish to use double precision (Look at the ODE_DIR/build and use premake to compile as required. This is true for ode 0.11 for later versions please read their updated instructions). ODE installation instructions are in the INSTALL.txt file (ODE source root).

Set the environemtn variable ODE_DIR to point to the directory where you unpacked the sources.

Note: in ode 0.11.1 , the premake command is:

```
premake4 vs2005
```

SDL

The new simulator requires the SDL (http://www.libsdl.org/) (Simple DirectMedia Layer) drawing libraries.

Linux

In Linux Debian/Ubuntu just install libsdl1.2-dev. No environment variables will be needed.

```
sudo apt-get install libsdl1.2-dev
```

Windows

- Get the sdl archive from:

```
http://eris.liralab.it/iCub/downloads/packages/windows
```

```
or directly the SDL-devel-1.2.13-VC8.zip from http://www.libsdl.org/download-1.2.php.
```

- Unzip this archive where you like.

Set the environment variable SDLDIR to point to the directory created by the unzip procedure (e.g. SDL-1.2.13). CMake needs this environment variable to locate the library.

Add SDLDIR/lib to the system path.

Note If you get an error such as "the application has failed to start because SDL.dll was not found" it means that you did not update correctly the system path (last point in the previous list).

Simulator

The latest version of the iCub simulator can be found in $ICUB_ROOT/main /src/simulators/iCubSimulation.

To know how to use the simulator refer to the README file included with the source code. (Also available in this manual as Simulator README).

To compile the simulator

If the required libraries are correctly installed, icub simulator will be compiled correctly when compiling iCub software. Make sure you execute the install_applications target.

To run the simulator run iCub_SIM, it will then pick up the default configuration files if make install_applications is used. Alternatively specify the configuration file on the command line.

Drawstuff dependencies

The dependency on these routines has now been removed. For old notes see here.

Retrieved from "http://wiki.icub.org/index.php?title=ICub_Simulator_Installation&oldid=14998"

UPMC iCub project/Arboris-Python

From Wiki for RobotCub and Friends

Contents

What is Arboris-Python

Arboris-Python is a dynamic simulator, developed and maintained by Joseph Salini. More information here (http://www.isir.upmc.fr/?op=view_profil&lang=fr&id=121&pageid=766) .

Installation

Linux

If you do not have them installed yet, download and install python and few related tools:

```
sudo apt-get install ipython python-setuptools python-sphinx graphviz
```

You can download the code of Arboris and its visualization tools on git, then enter each folder and launch the setup command, like specified in the following.

arboris-python: main simulator

```
git clone https://github.com/salini/arboris-python.git
cd arboris-python/
python setup.py install --user
```

pycollada: tool for read/write collada files. this is need to visualize the mechanical simulation

```
git clone https://github.com/pycollada/pycollada.git
```

```
cd pycollada/
python setup.py install --user
```

pydaenim: tool for visualize the graphical simulations

```
git clone https://github.com/salini/pydaenim.git
cd pydaenim/
python setup.py install --user
```

Installing with the --user option puts the executables on $HOME/.local/bin. You need to add this folder to the PATH, hence modify .bashrc to add this line:

```
export PATH=$PATH:$HOME/.local/bin
```

If you install without this option, everything will be installed in /usr/local - for some reasons this is not advisable.

To start playing with simulation, for iCub for example, you may want to look at the controllers developed by Joseph Salini, which run on Arboris.

```
git clone https://github.com/salini/LQPctrl.git
cd LQPctrl
python setup.py install --user
```

Windows

tbd

Mac

tbd

Documentation

Documentation can be browsed online here (http://chronos.isir.upmc.fr/~salini/arboris/documentation/) .

You can also compile the documentation locally on your machine using sphinx, with the following commands

```
cd arboris-python/
python setup.py build_doc
```

Html documentation can now be browsed in arboris-python/build/sphinx/html.

First steps

Notes

This page is under construction. Please ask Serena Ivaldi (http://chronos.isir.upmc.fr/~ivaldi/) for more info.

Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/Arboris-Python&oldid=17148"

UPMC iCub project/Starting iCub

From Wiki for RobotCub and Friends

Contents

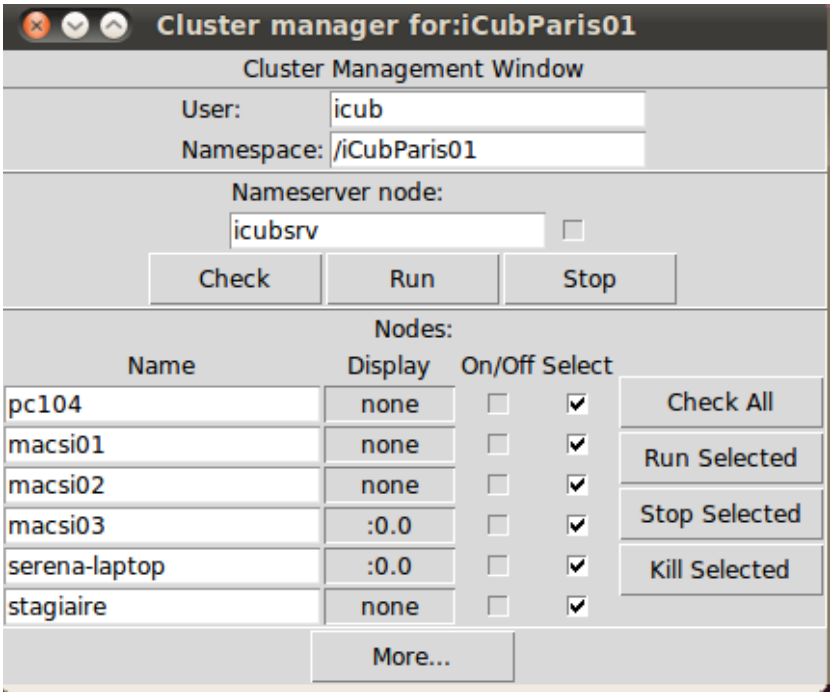
Start and calibration

To start using iCub and its cluster in ISIR, please read the following guidelines. In any case, ask Serena.

Starting yarp and pc104

- check first if **icubsrv** is running: it should be, there's an explicit "never shutdown this pc" warning on it
- start the desktop **macsi03** (the first black one next to the big table)
- give power to the table (green switch on the top right of the table)
- start the power supplies: top and bottom
- give power to the pc104 of the robot first, turning on the green switch named "pc104" (right): the pc104 boots
- wait for approximately 2 minutes, until the current consumption is around 1.8A (there is a peak of current during the boot of pc104)
- in the desktop of macsi03, click in the shortcut in the top bar, named "login to pc104": if asks for password, it is not ready yet. if it logins automatically without asking password, it's ready and we can proceed
- on macsi03, launch the script on the desktop:

```
sh Desktop/launchScript.sh
```



- look at the **icub cluster GUI**: if there are green ticks this means there's an old yarpserver or yarprun still working on; in this case, it is better to stop them and restart them all. hence, stop the machines first (selecting the corresponding checkboxes and then clicking "stop selected"). then as a precaution, open a terminal and perform a

```
yarp clean --timeout 0.8
```

to eventually clean pending ports. then stop the yarpserver clicking on its stop button.

- start **yarpserver** first by clicking on the corresponding start button
- in the same GUI, select pc104 and macsi03, and click on the button to launch **yarpruns**
- check if everything is green and working. as always, you can open a terminal and type

```
yarp detect --write
yarp name list
```

to check if servers have been opened.

Give power to the motors

- power the motors of the robot: press the red button (the "fault", aka mushroom red button) then turn on the green switch named "motors" (left)
- release the red button: it was just a precaution
- you can now launch the iCubInterface

Launching iCubInterface

From pc104:

- open a pc1404 terminal, using the shortcut provided in macsi03 (or the terminal opened previously), launch iCubInterface

```
iCubInterface --from FILE
```

which is generally

```
iCubInterface --from /usr/local/src/robot/iCub/main/app/robots/iCubParis01/conf/iCubInterface.ini
```

- if there's the case, you can dump the log (stderr and stdout) in a file on the pc104

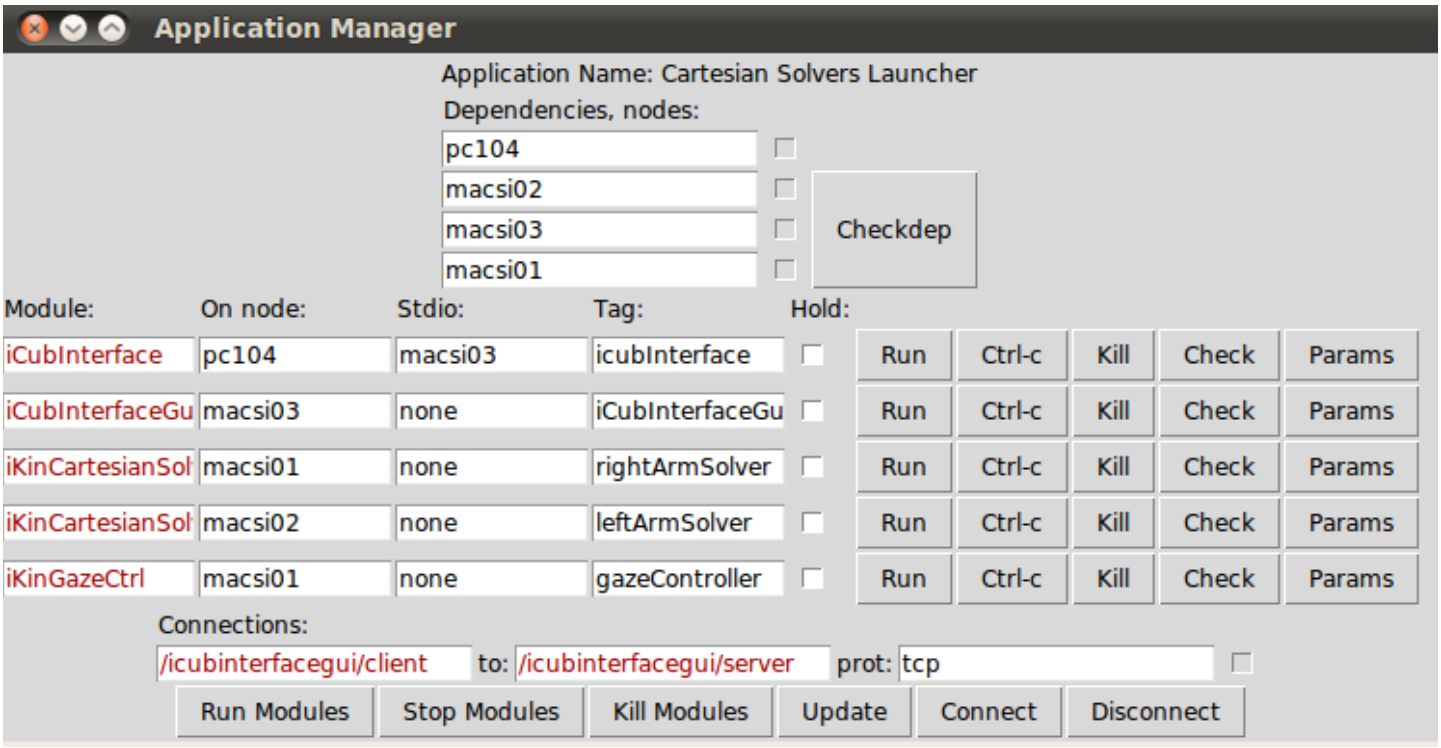
```
iCubInterface --from /usr/local/src/robot/iCub/main/app/robots/iCubParis01/conf/iCubInterface.ini >log.txt 2>&1
```

then copy it via scp, into macsi03 for example

```
scp icub@pc104:log.txt log.txt
```

Using the GUI:

- in the icub-app list, select the GUI called **CartesianSolver.xml**, then click on the iCubInterface. Eventually, you can click to start all modules.



Important!! once you launch the iCubInterface, pay attention to the calibration procedure: first the robot move the head and the arms, then closes fingers, opens fingers, then roll the head, closing the eyes. The head pitch is working fine only when the icub doesn't wear the head covers: this is a known issue of the platform. Thus, if you are calibrating with the head cover, you should experience some delay in the head pitch/yaw calibration and a little more noise in the pitch motor. Also some oscillations when going to

the zero pose of the head.

Important!! if some parts do not calibrate (e.g. known issue with left arm), please report it immediately to Serena.

Checking robot status

- check with the **robotMotorGui** if all interfaces to the joints are fine. open a terminal in macsi03 and type

```
robotMotorGui
```

all joints should be in position mode, that is a green tile for each joint. Check if all the encoders values are coherent with the values of the joints. If some joints are idle (yellow tile) or there's a mismatch, please report immediately to the administrator (Serena)

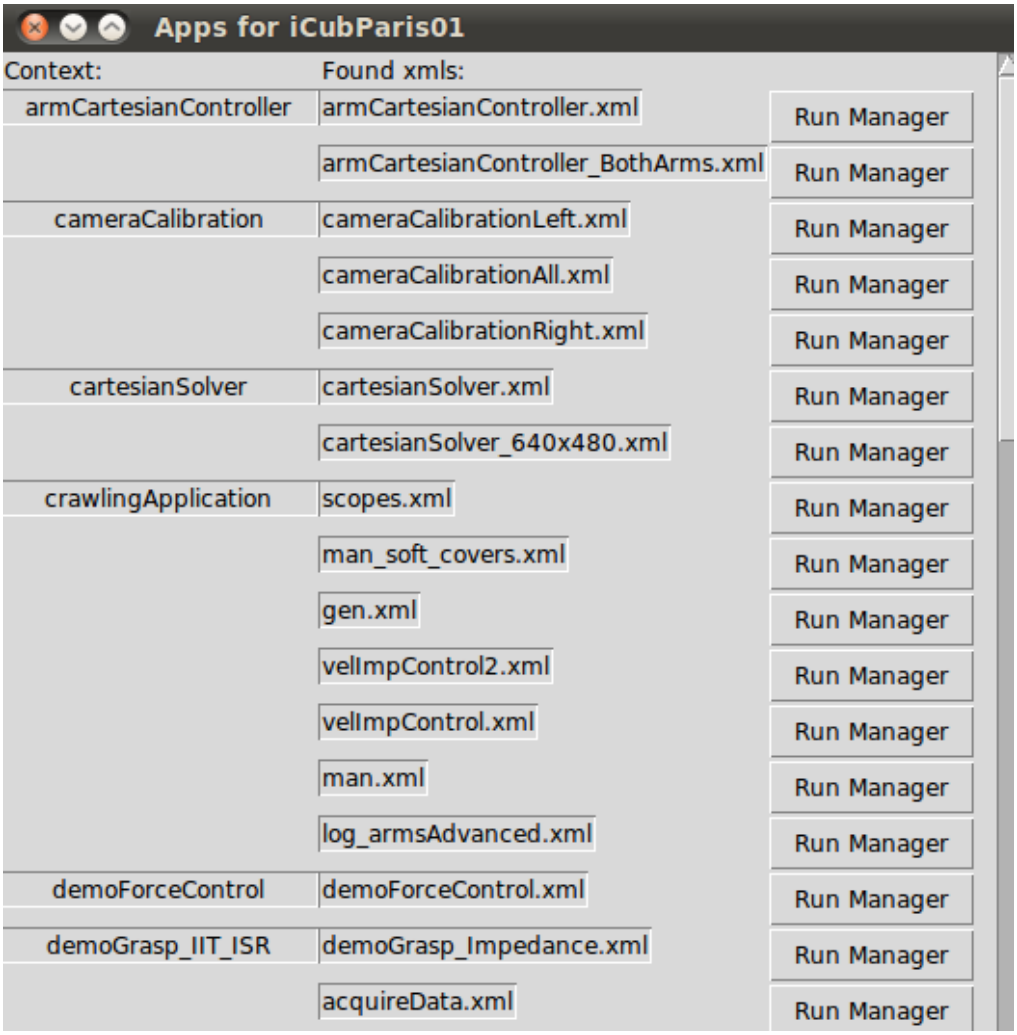
- if you didn't launch the **iCubGui** together with the iCubInterface (previous steps), in the icub-app list, select the GUI called **CartesianSolver.xml**, then click on the iCubGui, then "connect": you should check the status of all networks. if you see some asterisks it is normal: they should disappear once you launch the wholeBodyDynamics module. if they don't disappear, please report immediately to Serena
- you can now use the robot

Important! it is mandatory that there must be always one hand ready to push the emergency red button of the iCub: if something weird happens, if he's too fast in movements, if he collides with objects or self and is moving in position or velocity mode (i.e. impedance or torques are not enabled), then the red button must be immediately pushed. When doing this, the robot usually falls down like a stone, so there must be someone ready to "take" it and prevent a rough fall which can be dangerous.

Running modules

- You can easily use the GUIs to launch standard applications. They are started with the script:

```
sh Desktop/launchScript.sh
```



- To launch modules from your laptop, you must first connect your laptop to the iCub switch (16 doors, big one), the server dhcp will give you a dynamic IP 10.0.0.x. Then connect YARP to the iCub network, doing:

```
yarp namespace /iCubParis01
yarp conf 10.0.0.1 10000
```

Sometimes, it is just sufficient to do

```
yarp namespace /iCubParis01
yarp detect --write
```

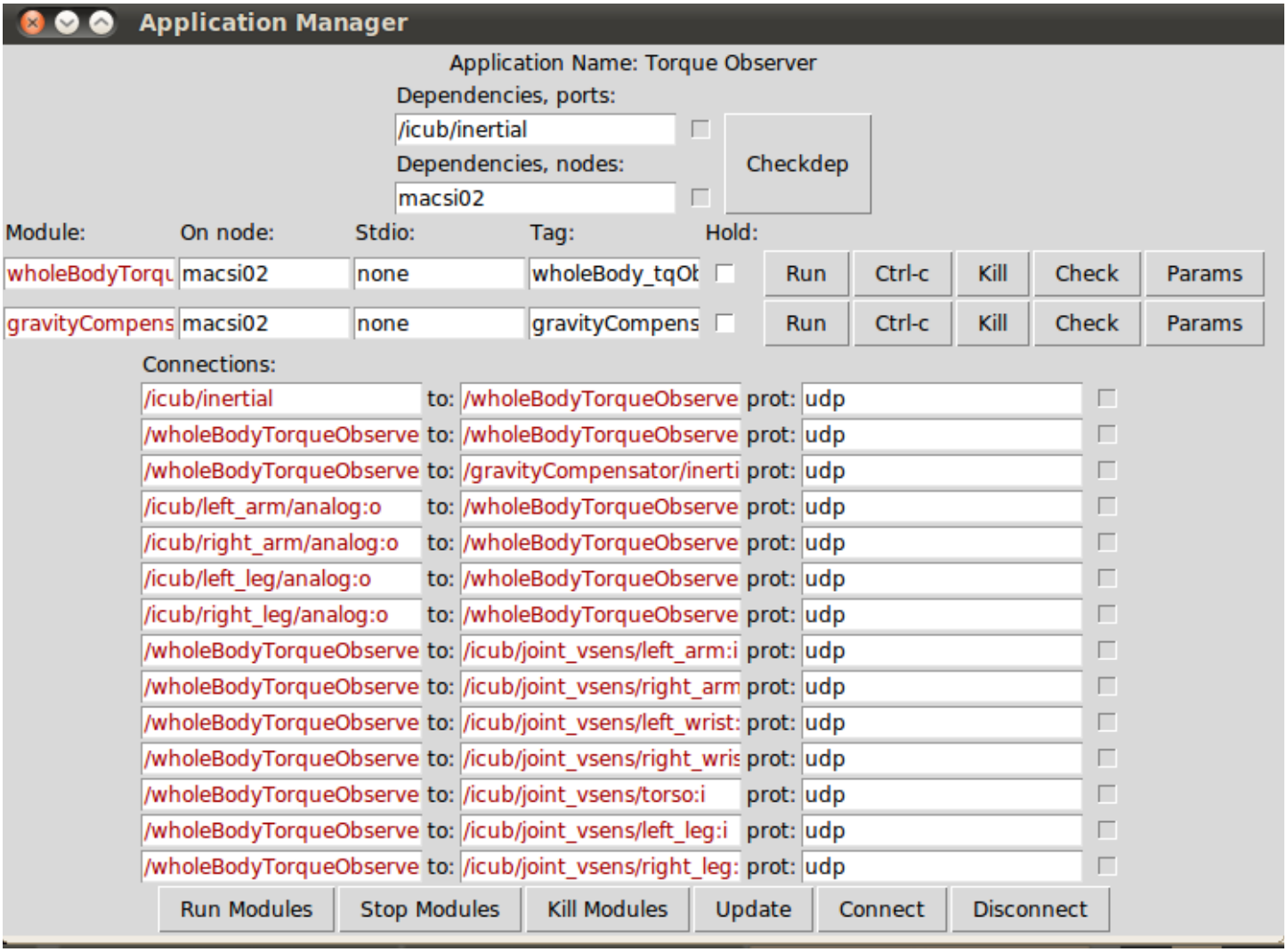
If you launch your modules using a GUI, remember to start your yarprun node only after connecting to the yarpserver

```
yarprun --server /myNode
```

Important!! since 10/2011 we are using yarpserver3, so be sure you have the most recent version of yarp in your pc if you are connecting to the icub from yours and not from one of the pc of the cluster!

Force control, impedance and compliance modules

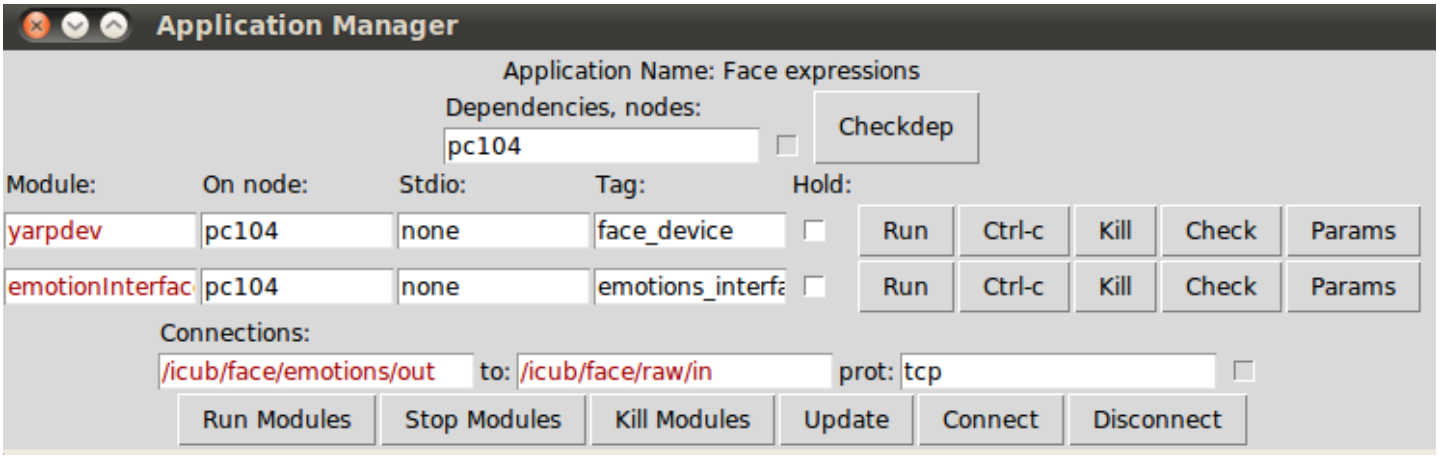
- if you are planning to do some compliant control, don't forget to start the **wholeBodyDynamics** module. in the app-list, choose the wholeBodyDynamics.xml GUI, then start the whole body module. wait for some seconds, click on "update", and wait for ports to become green/available. then click on "connect ports".



Face expressions

- you can enable face expressions by running the pc104 module commanding the led inside the iCub cover. Of course, if the cover is removed, this is useless. in the app-list, choose the FaceExpression.xml GUI then run and connect.
- in macsi03, on the Desktop, there are some pre-programmed sh scripts which cycle all the facial expressions

```
sh Desktop/cycleFacialExpressions.sh
sh Desktop/cycleFacialExpressionsHappy.sh
```



Retrieved from "http://wiki.icub.org/index.php?title=UPMC_iCub_project/Starting_iCub&oldid=15165"

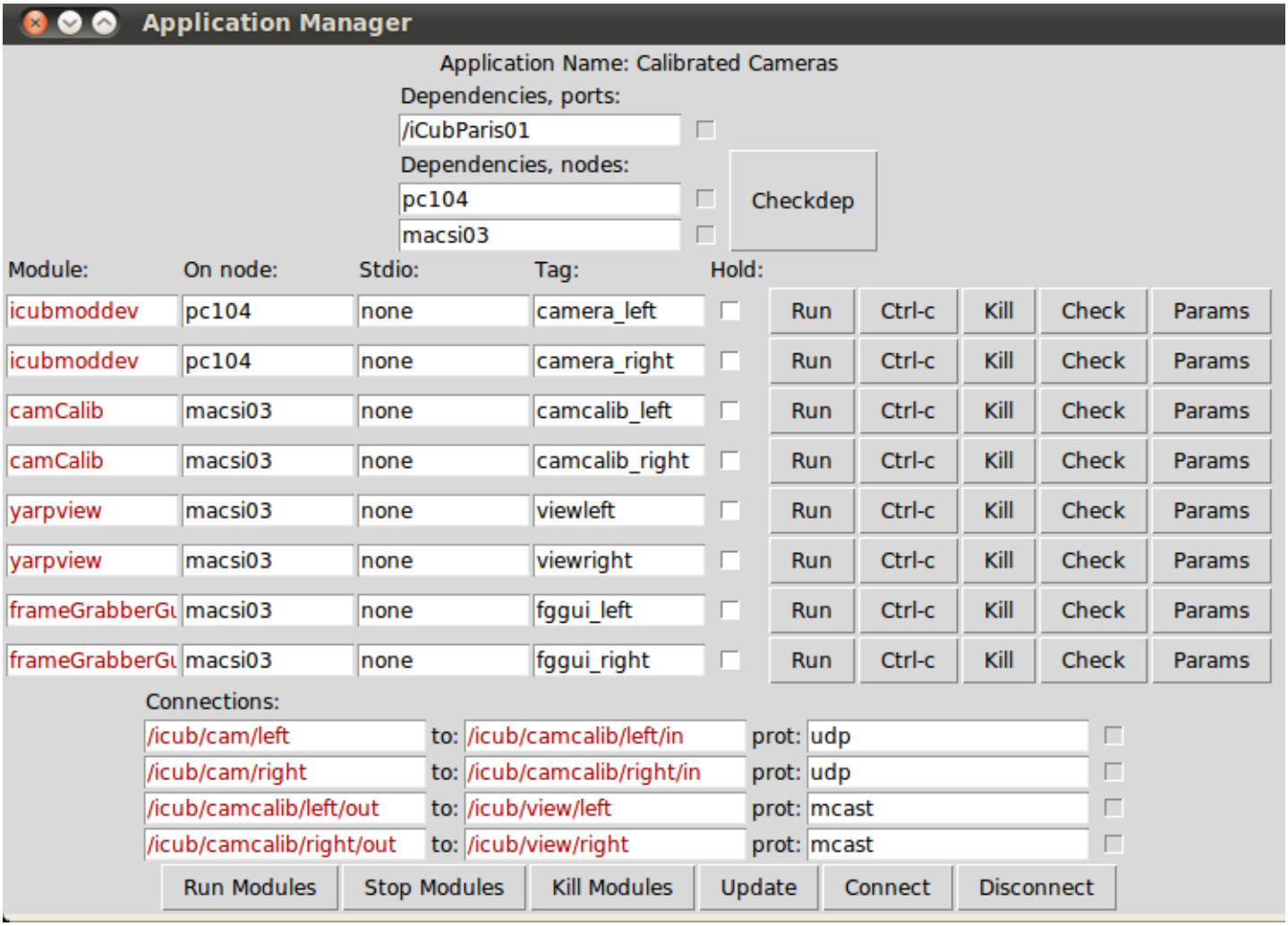
UPMC iCub project/cameras

From Wiki for RobotCub and Friends

Contents

Starting cameras

- Start iCub using the procedure described in the 'start iCub' page (http://eris.liralab.it/wiki/UPMC_iCub_project/Starting_iCub)
- in the icub-app list, select the GUI called **cameraCalibrationAll.xml**, then run all modules, and connect



Camera parameters

The cameras parameters can be set using the framegrabber GUI.

Suggested parameters for experiments and demos

- Left eye for Red-Ball demo

```
shutter     0.665
brightness  0.583
gain        0.413
exposure    0.027
sharpness   1.000
hue         0.480
saturation  0.436
gamma       0.400
iris        1.000
```

History of params

- 2011-06-30: experiments Serena & Jean-Philippe

```
shutter     0.876
```

```
brightness  0.373
gain        0.161
exposure    0.190
whitebal R  0.368
whitebal B  0.681
sharpness   0.945
hue         0.481
saturation  0.446
gamma       0.446
iris        0.554
format      640x480
fps         30
iso         400
```

BallTracker and 3D Reprojection

Start cameras

Start iCub and cameras with cameraCalibrationAll.xml (see above)

Launch BallTracker

Start balltracker.xml, run modules and connect

Several windows are created to tune parameters according to the ball to track.

- Window "Mask - Sat/Val" : tune parameters to keep the entire ball in white, while having the most of the background in black
- Window "Raw hue filter" : tune the hue being tracked to have the ball lighter than everything else (do not worry about the lights, they should be filtered by previous mask). Try to have the biggest contrast between ball and background
- Window "Pre-threshold" : tune to make most of background black and keep only the ball
- Window "Post-treshold" : adjust the threshold if you did not succeed in keeping only the ball visible in previous step

3D Reprojection

If you tuned correctly the balltrackers (for each camera), the 3D position of the ball should be broadcasted on /ReprojectionModuleName/pos:o